ONELAB: Open Numerical Engineering LABoratory

C. Geuzaine [1], F. Henrotte [2], E. Marchandise [2], J.-F. Remacle [2], P. Dular [1,3], R. V. Sabariego [1]

[1] Dept. of Electrical Engineering and Computer Science, University of Liège, Belgium
[2] Institute of Mechanics, Materials and Civil Engineering, Université Catholique de Louvain
[3] Fonds de la Recherche Scientifique - FNRS, Belgium
E-mail: cgeuzaine@ulg.ac.be

***Abstract* — We present the ONELAB software, a lightweight open source toolkit to interface finite element and related solvers used in a variety of engineering disciplines, and to construct multi-code models with maximum flexibility, efficiency and user-friendliness.**

I. INTRODUCTION

The industrial and the academic world share a global need for scientific computation software, in many domains from mechanical and electrical engineering to chemistry and biomedicine. While licensing costs for commercial tools are justified for large companies that use them extensively, we have witnessed first hand that smaller, more occasional users cannot justify the costs. Open source software constitutes an alternative: for scientific computing, professional quality codes of high scientific value are available in various engineering disciplines since the early 2000's: OpenFOAM [1] for computational fluid dynamics, Code_Aster for structural analysis [2], GetDP for electromagnetics [3], ... These codes are competitive when compared with their commercial counterparts, in regard to both their capabilities and their performance [4, 5].

However, these tools still have a marginal impact in small- and medium-size businesses, and in education. We believe that the main reason is their lack of a common, easy-to-use interface (for pre- and post-processing as well as for parameter input), together with nonexistent or scarce documentation and examples—at least for the codes originating from academia. Also, we believe that industry is still reluctant to adopt open source tools due to the ongoing confusion between "open source" and "limited, unprofessional" freeware.

This techno-economical analysis coalesces with the fact that product developers in industry need system-level simulation tools. This means tools with significant multi-physics capabilities, whereas specialized codes like OpenFOAM and Code_Aster remain essentially mono-physics. Existing platforms for multiphysics simulations offer solutions, both commercial (e.g. ANSYS Workbench [6] or COMSOL [7]) and open-source (e.g. SALOME [8] or Elmer [9]). However, the former are again expensive and the latter either lack the sought-after nimbleness and user-friendliness due to a "heavy-weight" top-down design, or lack the ability to interactively interface multiple specialized codes.

This led to our design goal for the ONELAB software library, directly inspired by (and based upon) the design of the open-source CAD modeler, mesh generator and post-processor Gmsh [10, 11]: create a fast, light and user-friendly interface to popular open source solvers in order to construct multi-code models with maximum flexibility and efficiency.

This work has been supported by the Belgian French Community (ARC 09/14-02) and the Walloon Region (WIST3 No 1017086 "ONELAB").

II. MULTI-CODE SIMULATIONS

Modelling is sometimes presented as the "art" of devising appropriate simplifications that allow formalizing a problem into a mathematical model that represents reality and remains nonetheless tractable in terms of complexity and computation time. Usually the purpose of those simplifications is to decouple weakly coupled phenomena. This pragmatic approach is associated with a loss in accuracy that must eventually be surpassed whenever accuracy requirements becomes more stringent. In that case, all or some of the conventional simplifying assumptions that justify the decoupling of the physics will be relaxed, and a multiphysics model is obtained.

Literature and the authors' experience show that the standard approach to multiphysics modelling by addition of extra functionality to a reference solver has severe limitations. The additional modules must be (at least partially) rewritten, and they remain therefore usually at a rather low level of sophistication when compared to their equivalents in specialized codes.

The alternative is to proceed by direct interfacing of specialized software rather than by the implementation of new functionalities in an existing code. This approach, which is also that of a platform like SALOME, allows using specialized simulation codes always with their latest and most advanced functionalities. The difference between SALOME and ONELAB resides in that the latter is designed as a lightweight toolkit rather than an integrated platform. To achieve this goal, the design of ONELAB is based on three main abstract interfaces.

III. ABSTRACT INTERFACES

ONELAB is based on a triple abstraction, which will be detailed in the extended paper:

- abstraction of the geometry modeling interface (CAD) together with the generation of finite element meshes;
- abstraction of the definition of physical properties, constraints and driving parameters of the target codes;
- abstraction of the post-processing layer.

The implementation is based on a client-server model, with a server-side database and (optional) graphical front-end, and local or remote clients communicating in-memory or through TCP/IP sockets. Contrary to most available interfaces, ONELAB has no *a priori* knowledge about any specifics (input file format, syntax, ...) of the simulation codes it calls. In practice, this is made possible by having any simulation preceded by a analysis phase, during which the clients are asked to upload their parameter set to the server. For clients that are linked with the ONELAB library (e.g. GetDP), the specification of which data to share is completely dynamic; for the others, ONELAB acts as a preprocessor of their input files, which should be instrumented

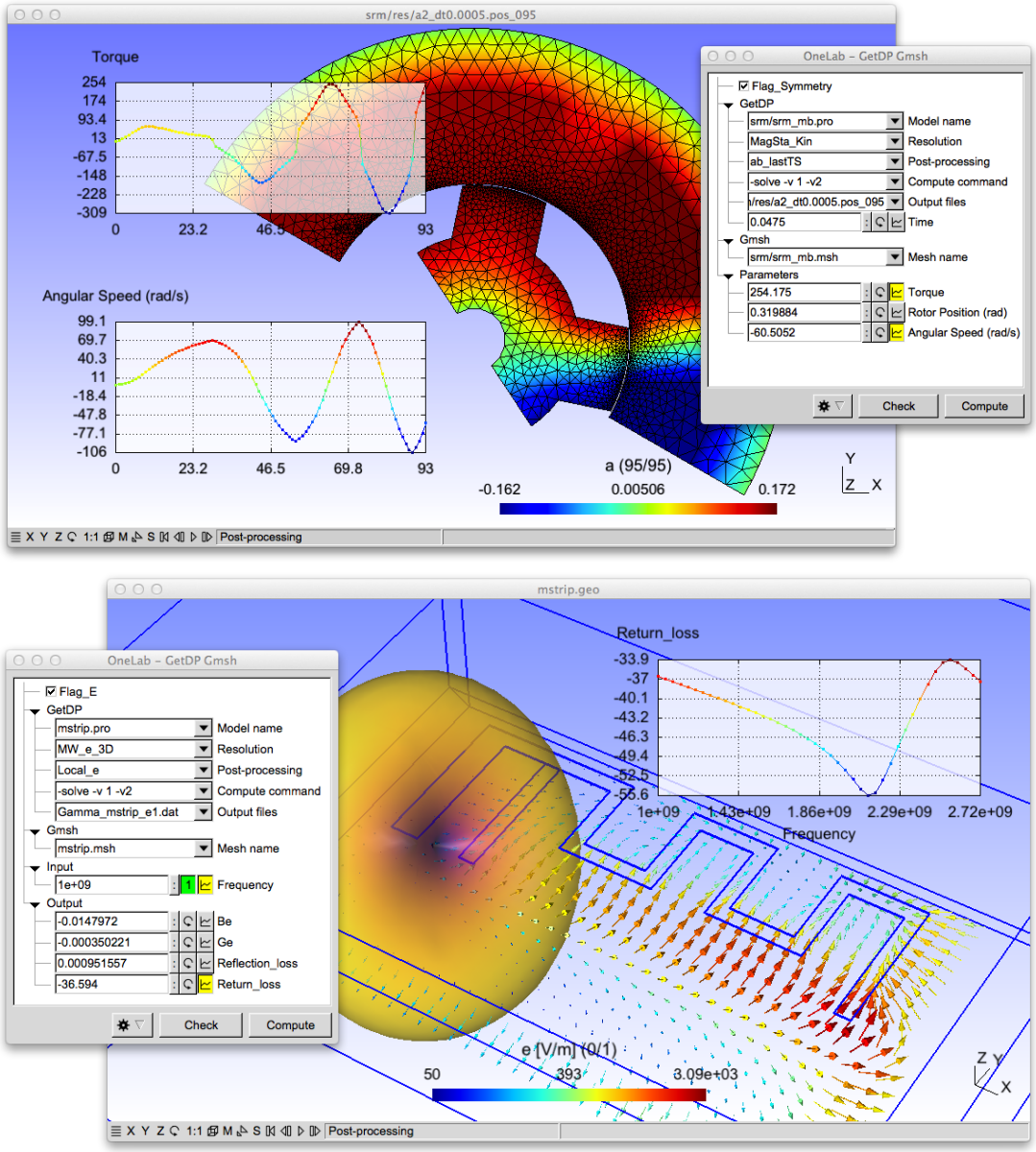

Fig. 1. Two examples of the graphical ONELAB front-end for parametric solver interaction in electromagnetics: a switched reluctance motor (top) and a microstrip antenna (bottom) simulated with Gmsh [10, 11] and GetDP [3].

to specify the information to be shared. The issues of completeness and consistency of the parameter sets are thus completely dealt with on the client side: the role of ONELAB is limited to data centralization, (optional) modification and redispatching. The same philosophy applies to the CAD and post-processing layers, which are both treated as yet another simulation code. Through the abstract Gmsh interface detailed in [11], a variety of CAD modelers can for example be used to create complex *native* parametric geometric models.

REFERENCES

[1] OpenFOAM, the open source CFD toolbox: `http://www.opencfd.co.uk/openfoam/`

[2] Code_Aster: `http://www.code-aster.org`

[3] GetDP, a General Environment for the Treatment of Discrete Problems: `http://www.getdp.org`

[4] G. Fernández, M. Meis and F. Varas. *Free software for numerical simulation: industrial applications*. Departamento de Matemática Aplicada II, Universidad de Vigo, Spain. XIII Spanish-French School Jacques-Louis Lions on Numerical Simulation in Physics & Engineering, Valladolid, 15-19 september 2008.

[5] A. Nilsson. *Some experiences on the accuracy and parallel performance of OpenFOAM for CFD in water turbines*. Lecture Notes in Computer Science, Volume 4699, p 168-176, 2009.

[6] Ansys Workbench, Platform for Advanced Engineering Simulation: `http://www.ansys.com/Products`

[7] COMSOL: `http://www.comsol.com`

[8] SALOME, the Open Source Integration Platform for Numerical Simulation: `http://www.salome-platform.org`

[9] Elmer, Open Source Finite Element Software for Multiphysical Problems: `http://www.elmerfem.org`

[10] Gmsh: `http://www.geuz.org/gmsh`

[11] C. Geuzaine and J.-F. Remacle, *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*. Int. J. Numer. Eng, 79(11), pp. 1309–1331, 2009.